Addressing Tokenization Inconsistency in Steganography and Watermarking Based on Large Language Models

Anonymous ACL submission

Abstract

Large language models have significantly enhanced the capacities and efficiency of text generation. On the one hand, they have improved the quality of text-based *steganography*. On the other hand, they have also underscored the importance of *watermarking* as a safeguard against malicious misuse. In this study, we focus on tokenization inconsistency (TI) between Alice and Bob in steganography and watermarking, where TI can undermine robustness. Our investigation reveals that the problematic tokens responsible for TI exhibit two key characteristics: **infrequency** and **temporariness**. Based on these findings, we propose two tailored solutions for TI elimination: a *stepwise verification* method for steganography and a *post-hoc rollback* method for watermarking. Experiments show that (1) compared to traditional disambiguation methods in steganography, directly addressing TI leads to improvements in fluency, imperceptibility, and anti-steganalysis capacity; (2) for watermarking, addressing TI enhances detectability and robustness against attacks.[1]

1 Introduction

Large language models (LLMs), such as GPT-3 (Brown et al., 2020), GPT-4 (Achiam et al., 2023), Gemini (Team et al., 2023, 2024), and Claude 3 (Anthropic, 2024), have revolutionized natural language processing and showcased impressive near-human-level text generation capabilities. These advanced LLMs facilitate the creation of flexible and contextually coherent text across diverse genres for text-based *steganography* (Yang et al., 2019; Ziegler et al., 2019) — a promising field in safeguarding information, referring to the art of concealing messages within texts.

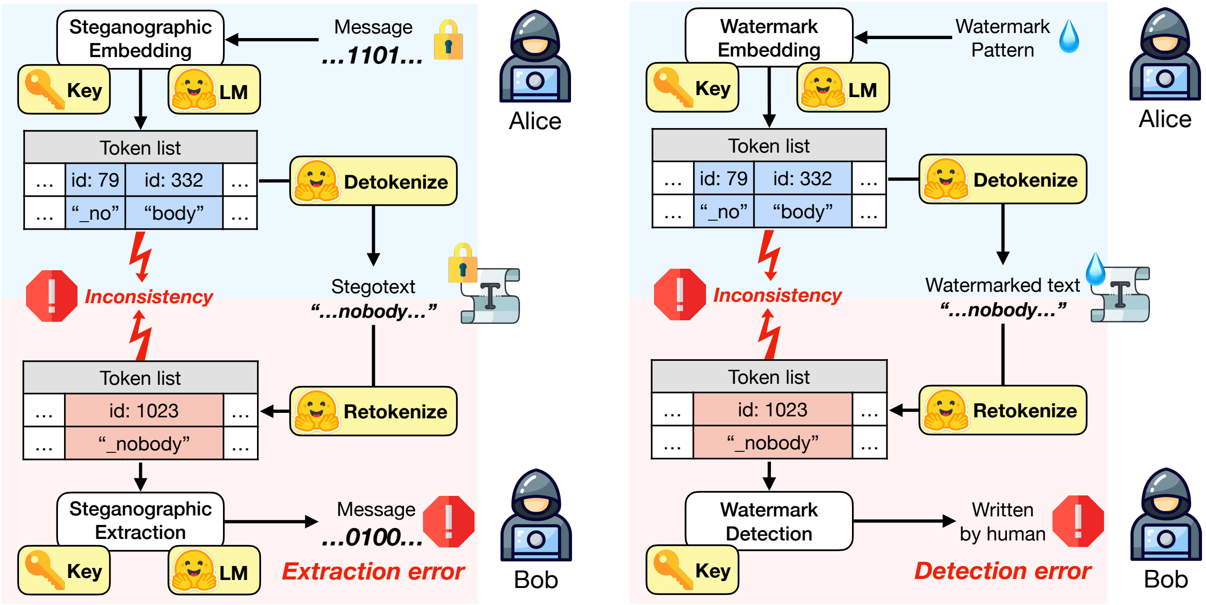
However, the same human-like text generation capabilities also pose risks, as synthesized content can be exploited for malicious purposes (Bergman et al., 2022; Mirsky et al., 2023). To address this, *watermarking* techniques for LLMs (Kirchenbauer et al., 2023; Zhao et al., 2024) have been developed, embedding imperceptible yet algorithmically detectable signals into generated text. These techniques play a crucial role in ensuring the detectability and responsible use of LLM-generated content.

In both steganography and watermarking applications, Alice (the sender) employs LLMs to generate steganographic texts (stegotexts) or watermarked texts, which are then transmitted to Bob (the receiver). During this process, an intermediate **detokenization-retokenization** pipeline is applied to the text as it moves from Alice to Bob. As a result, tokenization inconsistency (TI) (Sun et al., 2023) can arise, where discrepancies occur between the originally generated token list and the retokenized token list, potentially impacting the robustness of the system. Specifically, the inconsistent tokens generated by Alice which are responsible for TI are referred to source inconsistent tokens (SITs), while the corresponding inconsistent tokens resulting from Bob's retokenization are termed consequential inconsistent tokens (CITs). Figure 1 exemplifies how TI causes negative impacts on steganography (1a) and watermarking (1b).

Inconsistent tokens have not been systematically investigated in view of the detokenization-retokenization pipeline. Especially in steganography and watermarking, they comprise **robustness**, and can be **100% removable**. Motivated by these facts, this study aims to deepen the understanding of inconsistent tokens in both steganography and watermarking. Specifically, we achieve 100% correct extraction for steganography with minimal negative impact, and to enhance the detectability and robustness of LLM watermarks.[2] The key con-

[1] Anonymous code: https://anonymous.4open.science/r/AddressingGlitch-FEE9.

[2] Any inconsistent token could be catastrophic for most LLM-based steganographic approaches, as any one-step extraction error could cause a series of errors (Qi et al., 2025).



(a) Tokenization inconsistency (TI) in steganography.

(b) Tokenization inconsistency (TI) in watermarking.

Figure 1: An example of tokenization inconsistency (TI) in LLM-based steganography or LLM-based watermarking. Alice generates a token sequence corresponding to subwords "_no" and "body" (SITs) during steganography or watermark embedding. During transmission, the generated tokens are detokenized into the text "nobody". However, the receiver Bob retokenizes the text "nobody" as a single token "_nobody" (a CIT). This can lead to errors in steganography extraction or watermark detection.

tributions of this work are as follows:

1) We investigate the emergence of inconsistent tokens during token-by-token generation by language models and identify two key characteristics: **infrequency** and **temporariness**.

2) Taking advantage of the infrequency, we propose a *stepwise verification* method for steganography, maintaining 100% correct extraction.

3) Taking advantage of both the infrequency and temporariness, we propose a *post-hoc rollback* method for watermarking, which is a lightweight variant method.

4) Experiments are conducted across various language models, demonstrating the superiority of our methods: (1) In steganography, compared to the best baseline in each group, our stepwise verification method improves fluency (lowering perplexity by 14.12%), imperceptibility (lowering KL divergence by 47.86%), and anti-steganalysis capacity (lowering steganalysis accuracy by 3.53%) of steganographic texts (stegotexts). (2) For watermarking, our post-hoc rollback method overall enhances the detectability and robustness compared to TI-unaware watermarking.

2 Investigation: Inconsistent Tokens in Generation by Language Models

We investigate how and to what extent inconsistent tokens emerge during token-by-token generation by language models. This investigation serves as a fundamental basis for studying TI in most steganography and watermarking techniques.

We employ three language models — Llama-2-7b[3] (Touvron et al., 2023), Swallow-7b[4] (Fujii et al., 2024; Okazaki et al., 2024) and Qwen2.5-7b[5] (Team, 2024; Yang et al., 2024) — respectively with English, Japanese and Chinese contexts, to investigate the behavior of inconsistent tokens. Their tokenizers are all based on subwords. For each language model and for each specified number of generated tokens, we generate 1,000 text samples. Texts are produced token by token, with each token sampled using multinomial sampling (in single-track generation). The experimental setups for this section are detailed in Appendix D.1.

Text-level inconsistency rate: The rate at which TI appears. Table 1 presents the text-level inconsistency rates across various language models and

[3] https://huggingface.co/meta-llama/Llama-2-7b-hf
[4] https://huggingface.co/tokyotech-llm/Swallow-7b-hf
[5] https://huggingface.co/Qwen/Qwen2.5-7B

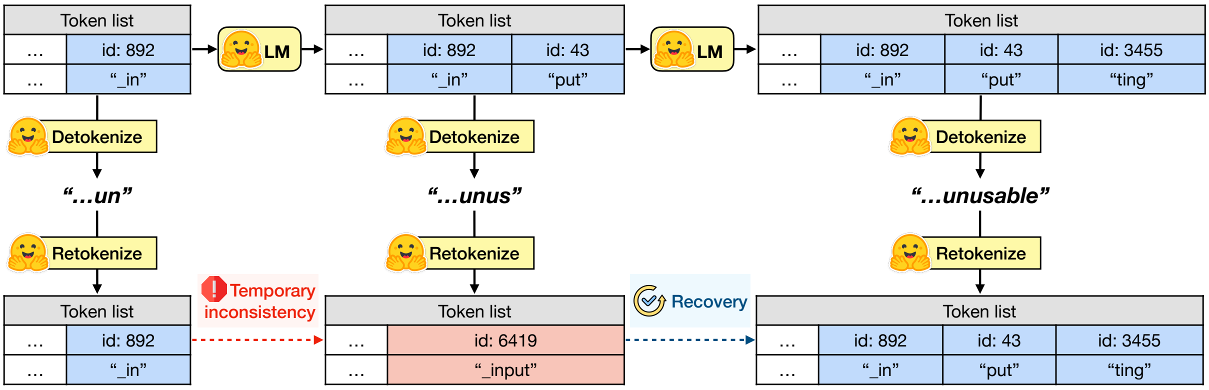


Figure 2: An example where a candidate-level IT causes TI which recovers back to consistency during token-by-token generation. As the token "put" (id: 43) can cause TI immediately after it is output, this token is a candidate-level IT. However, this token does not cause a long-term TI, as TI recovers afterwards.

Token number	Llama-2-7b	Swallow-7b	Qwen2.5-7b
25	3.9%	3.9%	8.7%
50	8.1%	5.0%	11.1%
100	17.7%	9.6%	18.1%
200	33.6%	15.5%	39.4%
400	53.0%	34.2%	66.4%

Table 1: **(Text-level inconsistency rates)** Rates of existence of TI.

Token number	Llama-2-7b	Swallow-7b	Qwen2.5-7b
25	0.176%	0.242%	0.558%
50	0.204%	0.181%	0.381%
100	0.215%	0.185%	0.349%
200	0.228%	0.157%	0.400%
400	0.223%	0.186%	0.467%

Table 2: **(Token-level inconsistency rates)** Ratios of the sum of SITs and CITs to the sum of generated tokens and retokenized tokens.

token generation lengths. Since each token carries some potential to become an inconsistent token, the general trend indicates that the incidence of inconsistent tokenization increases as the number of generated tokens grows.

Token-level inconsistency rate: The ratio of the sum of SITs and CITs to the sum of generated and retokenized tokens. According to Table 2, this metric is not closely related to text length. Another notable observation is that token-level inconsistency rates are typically below 0.5%, which results in much higher text-level inconsistency rates though. The disparity between high text-level inconsistency rates and low token-level inconsistency rates highlights *the infrequency of inconsistent tokens*.

	Llama -2-7b	Swallow -7b	Qwen2.5 -7b
Number ratio	1.497%	2.045%	3.993%
Probability ratio	0.184%	1.126%	1.833%

Table 3: **(Candidate-level inconsistency rates)** Number ratios and probability ratios of candidate-level ITs to tokens in candidate pools.

Candidate-level inconsistency rate: The ratio of candidate tokens that can cause TI to all tokens in candidate pools of token-by-token generative steps. To further explore the cause of the infrequency of inconsistent tokens, potential TI from candidate pools are investigated. How to determine if a candidate token is a candidate-level inconsistent token (candidate-level IT) is shown in Algorithm 1, which refers to a detokenization-retokenization pipeline in one step. For accurate calculation, those scenarios where TI has occurred before outputting a token are excluded.

For simplicity, only the 64 highest probability tokens in each candidate pool are considered. Table 3 respectively lists the number ratios and probability ratios of candidate-level ITs (the ratios of the cumulative numbers or probabilities of candidate-level ITs to the cumulative numbers or probabilities of top-64 tokens) across various language models, with data aggregated across various text lengths. The results indicate that the infrequency of inconsistent tokens is primarily due to the low candidate-level inconsistency rates in each candidate pool. The infrequency of the SITs is also revealed.

However, when observing the data in Tables 1, 2, and 3, another question arises: How does Llama-

Llama-2-7b	Swallow-7b	Qwen2.5-7b
8.76%	81.98%	87.93%

Table 4: **(Temporary inconsistency rates)** Rates of candidate-level ITs that do not cause TI in final among all candidate-level ITs.

2-7b, despite having a lower candidate-level IT probability ratio in candidate pools (0.184% vs. 1.126% in Swallow-7b), exhibit higher inconsistency rates at both the text level (17.7% vs. 9.6% with 100 tokens) and the token level (0.215% vs. 0.185% with 100 tokens)? Considering limitations in defining the candidate-level inconsistency rate, it is likely that generating candidate-level ITs may only result in *temporary* TI. Figure 2 instantiates this phenomenon.

Temporary inconsistency rate: Among the candidate-level ITs that are output, this metric represents the rate of temporary SITs (that do not cause TI after the entire generation process). Table 4 lists the temporary inconsistency rates for the texts generated by the three language models. The significantly lower temporary inconsistency rate for Llama-2-7b compared to others indicates that its generated potential SITs are more stable in affecting the final tokenization. This stability leads to higher text-level and token-level inconsistency rates than those observed for Swallow-7b.[6]

In summary, our investigation reveals that inconsistent tokens generated by language models are characterized as *(1) infrequency* and *(2) temporariness*. These findings inspire us to develop methods to address inconsistent tokens and TI in LLM-based steganography and watermarking.

3 Methods

In this section, the introduced methods are targeted to the mechanisms of LLM-based steganography and watermarking respectively, meanwhile taking advantage of the infrequency and temporariness of inconsistent tokens. Specifically:

For steganography: We propose a stepwise verification method that precisely removes candidate-level ITs at each generation step. As only outputting candidate-level ITs can cause (at least temporary) TI, the presence of candidate-level ITs is a necessary condition for the eventual occurrence of inconsistent tokens.[7] Therefore, the absence of candidate-level ITs is a sufficient condition for the final absence of inconsistent tokens. Hence, our method eliminates TI in the final output.

In steganography, since candidate tokens are associated with codewords, it is necessary to call the tokenizer to verify whether each candidate token is a candidate-level IT. Although this process may appear inefficient, our method operates with linear complexity, still providing some advantages over the previous disambiguation algorithms (Nozaki and Murawaki, 2022; Yan et al., 2023; Qi et al., 2025) whose complexities are at least $O(n^2)$. Both our method for addressing TI and previous disambiguation approaches share the same goal: ensuring 100% correct steganographic extraction.

Additionally, our method preemptively removes candidate-level ITs to achieve extraction accuracy. Since both candidate-level ITs and final inconsistent tokens are infrequent, KL divergence between the original and the modified candidate pools (resulting from removing candidate tokens) remains small, greatly reducing the negative impact on imperceptibility.[8]

For watermarking: We propose a post-hoc rollback method that makes the generation process rollback to the state where TI has not happened if TI persists. The rollback mechanism does not respond immediately to TI because of their temporariness. Unlike the method used for steganography, there is a higher relaxation for watermarking, because the detector does not require detailed information of candidate pools at each step to decode the text, so that candidate tokens do not need to be examined individually. Negative effects on imperceptibility can be nearly negligible because of the infrequency of the output inconsistent tokens.

3.1 A Stepwise Verification Method for Steganography

The overview of the stepwise verification method is shown in Figure 3, where the verification mechanism is placed between the sampling and steganographic encoding steps. Both the sender and receiver can verify whether each token in the candidate pool is a candidate-level IT, enabling them to perform steganographic encoding on the

[6] When using default tokenizer parameters, a fair amount of '<s>' and '</s>' output by Llama-2-7b lead to stable TI. Besides, Swallow-7b and Qwen2.5-7b are featured by outputing a fair amount of partial UTF-8 tokens (Land and Bartolo, 2024). Appendix F provides supplements for it.

[7] Detailed analysis and explanations about SITs, CITs, TI and candidate-level ITs are shown in Appendix B.

[8] Theories on imperceptibility are provided in Appendix A.2.3.

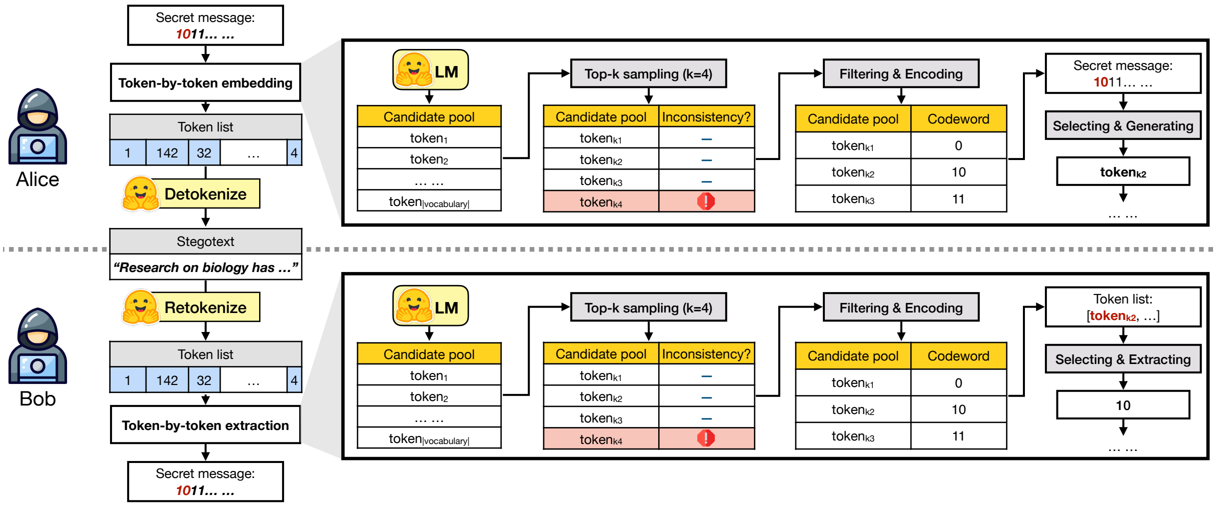
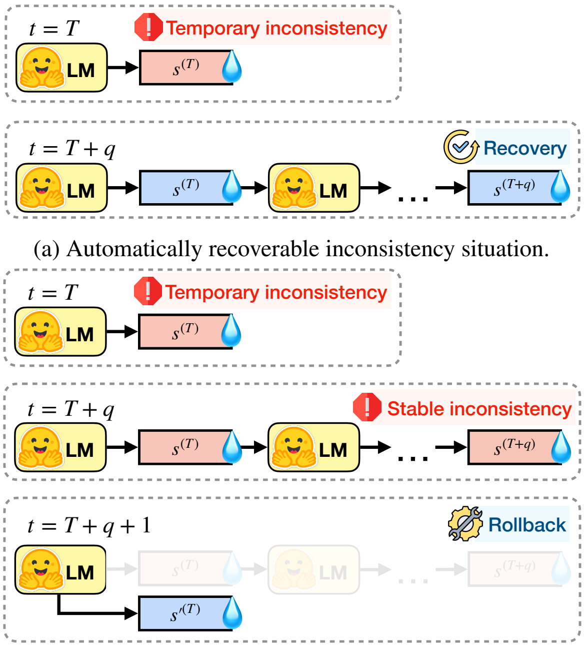


Figure 3: Overview and procedures of LM-based steganography with our stepwise consistency-verification approach. For some simplicity in this example, Huffman encoding (Yang et al., 2019) and top-4 sampling in each candidate pool is adopted. The verification mechanism is in place before encoding for both Alice and Bob.

```
Algorithm 1 Identify a candidate-level IT
Input:
Token to be verified, $s_o$
Previously generated token list, $L$
Output:
Candidate-level IT or not (True or False), $\text{Result}_G$

1: Append $s_o$ to $L$ to obtain $L_o$.
   /* Token list to be verified*/
2: Detokenize $L_o$ into a temporary text $t_{temp}$.
3: Tokenize $t_{temp}$ into $L'$.
4: $\text{Result}_G \leftarrow \neg(L_o == L')$;
5: return $\text{Result}_G$
```



(a) Automatically recoverable inconsistency situation.

(b) Stable inconsistency situation which needs rollback.

Figure 4: Mechanisms of the post-hoc rollback method. Due to temporariness of inconsistent tokens, a q-token observation period is set for them.

same filtered candidate pools. This guarantees that the receiver can accurately extract the secret messages. How to identify a candidate-level IT from a candidate pool is detailed in Algorithm 1.

The process simulates detokenization and retokenization of the generated stegotext transmitted from Alice to Bob. candidate-level ITs are removed from the candidate pool, as they could interfere Bob's extraction process. By eliminating such problematic tokens, the approach ensures that both Alice and Bob maintain identical token sequences, enabling reliable steganographic extraction.[9]

3.2 A Post-Hoc Rollback Method for Watermarking

This method of removing inconsistent tokens for LLM watermarking leverages not only the infrequency of inconsistent tokens but also their temporariness. The overview of this post-hoc method is illustrated in Figure 4, which only involves the token-by-token watermark embedding.

The core idea is as follows: If a candidate-level IT is generated and causes temporary TI at that step, the token is assigned an *observation period* that lasts until the next q tokens are generated. Once the observation period ends, if tokenization consistency is recovered (Figure 4a), no further action is necessary. However, if TI persists, this temporary TI is deemed a stable TI, and the generation

[9] Further details can be found in Appendix C.1.

process rolls back to the state before the candidate-level IT was generated. At this regressed point, the candidate pool is resampled, excluding that candidate-level IT (Figure 4b).[10]

4 Experiments

In this section we explore the behaviors of addressing TI in steganography and watermarking, using Llama-2-7b (Touvron et al., 2023) (with English contexts), Swallow-7b (Fujii et al., 2024; Okazaki et al., 2024) (with Japanese contexts) and Qwen2.5-7b (Team, 2024; Yang et al., 2024) (with Chinese contexts) (the same as Section 2). The prompts are randomly selected from the multilingual C4 dataset (Raffel et al., 2019). Implementation details can be found in Appendix D.1.

4.1 Experiments on Steganography

The secret message for embedding is a random 128-bit message, i.e. $m \sim \text{Uniform}(\{0,1\}^{128})$. As our proposed stepwise verification method and those disambiguation methods (Section A.2.2) all enable steganographic extraction 100% correct, it is reasonable to use these 100% disambiguation algorithms as baseline methods, namely, Basic (Nozaki and Murawaki, 2022), MWIS (Yan et al., 2023), and SyncPool (Qi et al., 2025).

All methods employ arithmetic coding (Ziegler et al., 2019), an efficient attempt to provably secure steganography (Ding et al., 2023). To evaluate performance under varying embedding capacities, experiments are conducted with different top-k sampling values ($k \in \{4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096\}$). For each top-$k$ value, for each disambiguation method and ours, and for each language model, 500 samples are generated.

4.1.1 Metrics

Bits per token (**BPT**) is a fundamental metric in linguistic steganography, measuring the embedding capacity. Perplexity (**PPL**) assesses the quality and fluency of the generated text. KL divergence (**KLD**) between modified and original candidate pools quantifies statistical disparities, reflecting imperceptibility. Steganalysis accuracy (**ACC**) indicates the ability to invalidate detection, which is evaluated using a fine-tuned discriminator, with further details provided in Appendix D.2. Finally, the running time (**Time**, in seconds) to embed a secret message indicates the steganographic efficiency.

4.1.2 Results

While only 128 bits are embedded, there are non-negligible extraction error rates for all the three adopted language models, which are about 10% for Llama-2-7b, about 5% for Swallow-7b, and about 7% for Qwen2.5-7b (details are shown in Appendix D.3). This illustrates the necessity of our stepwise verification method or disambiguation methods for steganography.

For each method, average experimental data obtained under various top-k values are grouped into embedding-capacity intervals ($2.0 \leq$ **BPT** < 6.0). Tables 5, 6, and 7 show the average performance across these intervals for each approach. Note that when the sample size in any group is 20 or fewer, the data is considered insufficient and marked as “–” in these tables. For each group, within these valid data, the best data point is marked in green background and the worst data point is marked in red background. The main findings via these experiments are as follows:

1) One of the baseline methods, SyncPool (Qi et al., 2025), severely suffers from limitations in embedding capacity. Especially in Llama-2-7b and Swallow-7b, only when **BPT** < 3.0, there are sufficient data. Details and explanations of this problem are shown in Appendix D.4.

2) The operation efficiency of our stepwise verification method is acceptable. In our method, it is necessary to call the tokenizer to check each candidate token. Compared to baseline methods whose time complexity is at least $\mathrm{O}(n^2)$ (n is k in the case), the lightweight aspect of our complexity is $\mathrm{O}(n)$. According to the **Time** dimension, our method is much more efficient than MWIS (Yan et al., 2023). Even though the Basic (Nozaki and Murawaki, 2022) method is the most efficient, the gap between it and ours could be narrowed as **BPT** increases essentially as k increases.[11]

3) Overall, our method outperforms the baselines. Compared to the best baseline method for each metric for each language model in each interval, ours achieves an average reduction of 14.12% in **PPL**, 47.86% in **KLD**, and 3.53% in **ACC**.

4.2 Experiments on Watermarking

We implement our proposed post-hoc rollback method for two types of LLM watermarking, which are, respectively, (1) logit-based watermarking and

[10] More detailed operation steps for this method for watermarking are shown in Appendix C.2.

[11] How **BPT** varies as k varies can be found in Figure 5 in the Appendix.

	$2.0 \leq$ BPT < 3.0				$3.0 \leq$ BPT < 4.0			
	PPL↓	KLD↓	ACC↓	Time↓	PPL↓	KLD↓	ACC↓	Time↓
Basic	17.72	1.005	0.936	1.62	38.96	1.355	0.976	1.43
MWIS	9.13	0.138	0.784	6.33	16.37	0.149	0.797	11.40
SyncPool	10.73	0.154	0.853	3.04	–	–	–	–
Stepwise verification	8.69	0.070	0.766	5.35	15.07	0.031	0.846	8.09

	$4.0 \leq$ BPT < 5.0				$5.0 \leq$ BPT < 6.0			
	PPL↓	KLD↓	ACC↓	Time↓	PPL↓	KLD↓	ACC↓	Time↓
Basic	77.91	1.690	0.963	2.73	134.87	1.934	0.922	7.21
MWIS	28.64	0.187	0.625	12.04	–	–	–	–
SyncPool	–	–	–	–	–	–	–	–
Stepwise verification	26.41	0.022	0.621	9.10	47.73	0.020	0.704	9.51

Table 5: Quantitative comparison with previous disambiguation methods on Llama-2-7b and English contexts.

	$2.0 \leq$ BPT < 3.0				$3.0 \leq$ BPT < 4.0			
	PPL↓	KLD↓	ACC↓	Time↓	PPL↓	KLD↓	ACC↓	Time↓
Basic	18.18	0.834	0.952	1.90	36.60	0.929	0.961	3.62
MWIS	9.49	0.126	0.866	6.36	18.70	0.132	0.915	12.12
SyncPool	12.70	0.172	0.890	3.40	–	–	–	–
Stepwise verification	9.24	0.081	0.865	4.89	18.03	0.052	0.911	7.26

	$4.0 \leq$ BPT < 5.0				$5.0 \leq$ BPT < 6.0			
	PPL↓	KLD↓	ACC↓	Time↓	PPL↓	KLD↓	ACC↓	Time↓
Basic	73.11	0.995	0.984	6.04	130.71	0.985	0.948	7.32
MWIS	37.85	0.157	0.861	13.35	–	–	–	–
SyncPool	–	–	–	–	–	–	–	–
Stepwise verification	34.80	0.041	0.922	8.59	69.81	0.041	0.872	9.25

Table 6: Quantitative comparison with previous disambiguation methods on Swallow-7b and Japanese contexts.

	$2.0 \leq$ BPT < 3.0				$3.0 \leq$ BPT < 4.0			
	PPL↓	KLD↓	ACC↓	Time↓	PPL↓	KLD↓	ACC↓	Time↓
Basic	24.78	0.975	0.670	1.65	54.64	1.044	0.760	1.56
MWIS	12.17	0.185	0.625	4.28	27.00	0.175	0.762	7.87
SyncPool	17.55	0.105	0.615	4.73	41.03	0.094	0.755	4.98
Stepwise verification	12.07	0.152	0.597	4.15	24.76	0.111	0.695	5.18

	$4.0 \leq$ BPT < 5.0				$5.0 \leq$ BPT < 6.0			
	PPL↓	KLD↓	ACC↓	Time↓	PPL↓	KLD↓	ACC↓	Time↓
Basic	113.02	1.122	0.796	2.69	232.97	1.216	0.806	5.03
MWIS	53.70	0.188	0.758	9.75	111.11	0.193	0.686	13.93
SyncPool	85.36	0.088	0.750	6.53	–	–	–	–
Stepwise verification	50.08	0.093	0.699	6.68	102.78	0.075	0.667	8.25

Table 7: Quantitative comparison with previous disambiguation methods on Qwen2.5-7b and Chinese contexts.

(2) sampling-based watermarking.

For logit-based watermarking, the LeftHash scheme (Kirchenbauer et al., 2023) (the context width is 1), the SelfHash scheme (Kirchenbauer et al., 2024) (the context width is 4), and the Unigram scheme (Zhao et al., 2024) are adopted. For each method, we set green list size $\gamma = 0.5$, and hardness parameter (priority in logits of green list) $\delta = 2.0$. For sampling-based watermarking, the Gumbel softmax scheme (Aaronson and Kirchner, 2023)[12] is adopted (the context width is 5). Detailed schemes are shown in Appendix D.5.

Due to differences in temporary inconsistency rates between various language models (shown in Table 4), and this rate in Llama-2-7b is much lower than those of the other two models. The observation period q is also set differently according to these models, i.e. larger q representing higher relaxation, is set for scenarios with higher temporariness. Specifically, $q = 2$ is set for Llama-2-7b, and $q = 10$ is set for Swallow-7b and Qwen2.5-7b.[13] Besides, the number of generated tokens is constantly 200. Regardless of whether to use our post-hoc rollback method, 500 samples are generated and collected for each watermarking method and each language model.

[12] We use the version provided by Fu et al. (2024).

[13] How to determine q is detailed in Appendix F.

Watermarking scheme		**Unattacked**			**Attacked** ($\epsilon = 0.2$)		**Attacked** ($\epsilon = 0.4$)	
		Watermark Strength↑	AUROC↑	PPL↓	Watermark Strength↑	AUROC↑	Watermark Strength↑	AUROC↑
LeftHash	Original	7.58	0.996	20.55	4.59	0.982	2.57	0.878
	Post-hoc rollback	**7.73**	0.996	**19.56**	**4.82**	**0.984**	**2.58**	**0.879**
SelfHash	Original	7.33	0.999	20.53	4.09	0.967	2.01	0.812
	Post-hoc rollback	**7.44**	0.999	**19.87**	**4.30**	**0.973**	**2.05**	**0.820**
Unigram	Original	7.76	**0.998**	19.00	6.43	0.987	5.18	**0.912**
	Post-hoc rollback	**7.77**	0.995	**17.85**	**6.50**	0.987	**5.23**	0.910
Gumbel	Original	21.43	0.952	3.47	13.93	0.918	8.64	0.843
	Post-hoc rollback	**23.60**	**0.956**	**3.15**	**15.29**	**0.935**	**9.62**	**0.880**

Table 8: Quantitative comparison in various watermarking schemes on Llama-2-7b and English context.

4.2.1 Attacking Watermarks

Considering the scales, fairness, and availability of attacking models, we adopt the original language model that generates watermarked texts as the replacement model to attack watermarks.[14] For each token, it can be selected and then replaced by inference (according to the left context) and resampling, where the selection probability is ϵ. Besides, for more practical scenarios, results under GPT-4o paraphrasing attack are shown in Appendix H.

4.2.2 Metrics

The detectability of watermarked texts is denoted by **watermark strength**, where a higher watermark strength increases the likelihood of the text being detected as watermarked. Details about how to calculate the strength are shown in Appendix D.5. Due to differences in approaches to computing watermark strength, the obtained watermark strengths indicate relative scores and are not comparable across different watermarking types. Besides, **AUROC** value is employed to simulate detectability in real-world scenarios, where 500 watermarked texts and 500 unwatermarked texts are evaluated in each group. Perplexity (**PPL**) assesses the text quality.

4.2.3 Results

Table 8 lists the average experimental statistics in various watermarking methods under Llama-2-7b.[15] The main findings are:

1) The watermark strengths with the rollback mechanism exhibit a steady increase compared to the original. This aligns with the fact that inconsistent tokens interfere with the detection process. However, the extend of improvement is limited due to the infrequency of inconsistent tokens.

2) Some superiority of our post-hoc rollback method in watermark strengths and AUROC values under attack scenarios represents that watermarks after addressing TI are more detectable and more robust, i.e. higher anti-modification capability.

3) **PPL** in each group counterintuitively decreases steadily, as our method does not target it.[16] One possible explanation for this could be: The tokenizer-based perplexity calculation is affected by inconsistent tokens. Specifically, the predicted probabilities of CITs could be considered very low, thus **PPL** becomes higher finally.[17]

5 Conclusion

We observed that tokenization inconsistency (TI) in LLM-based steganography and watermarking can cause robustness issues in extraction or detection processes. Our investigation on inconsistent tokens across different LLMs and language contexts reveals the infrequency and temporariness of inconsistent tokens. Based on these two characteristics, we propose two methods to address TI: one for steganography and the other for watermarking. Our experiments, conducted across various language models, demonstrate that: (1) for steganography, our stepwise verification method outperforms traditional disambiguation approaches across embedding-capacity intervals, offering superior text quality, imperceptibility, and anti-steganalysis capacity; (2) for watermarking, our post-hoc rollback method enhances both detectability and robustness against adversarial modifications while maintaining lower perplexity. Our proposed methods have great potential in generalizability, as they can be applied to a wide range of steganographic algorithms and watermarking schemes.

[14] For watermarking attacks, the common T5 model (Raffel et al., 2020) does not support Japanese or Chinese.

[15] For Swallow-7b and Qwen2.5-7b, the results are shown in Tables 14 and 15 (in the Appendix).

[16] Perplexities in attack scenarios are shown in D.6.

[17] How inconsistent tokens affect perplexity is detailed in E.3.

Limitations

For steganography: Similar to mainstream linguistic steganographic approaches, the threat model assumes the absence of an active attacker capable of modifying the stegotexts. Otherwise, the guarantee of 100% correct steganographic extraction would not be ensured.

For watermarking: Compared to original TI-unaware watermarking schemes, the superiority of the watermarking schemes equipped with the post-hoc rollback method is limited. It is the infrequency of inconsistent tokens that makes improvements to various metrics relatively minor. Specifically, for comparison, any inconsistent token in steganography can disturb subsequent inference.[18] The inconsistent tokens in watermarking can merely influence scores at inconsistency positions and positions whose watermarking contexts contain inconsistent tokens during watermarking detection process.[19]

Ethical Considerations

Intended applications of steganography are embedding copyright information, countering censorship, and similar uses. However, it can also be used to be exploited for harmful purposes, such as covert communication by malicious actors, spreading disinformation, or bypassing censorship mechanisms. Hence, its potential to facilitate illicit activities necessitates robust monitoring and regulation to prevent misuse. In addition, countermeasures against steganography, steganalysis, the study of detecting the presence of hidden messages, would also be an encouraging research direction to safeguard against malicious use.

References

S. Aaronson and H. Kirchner. 2023. Watermarking of large language models. Watermarking gpt outputs. Technical report, openai.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku. `https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf`.

A. Stevie Bergman, Gavin Abercrombie, Shannon Spruit, Dirk Hovy, Emily Dinan, Y-Lan Boureau, and Verena Rieser. 2022. Guiding the release of safer E2E conversational AI through value sensitive design. In *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 39–52, Edinburgh, UK. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Miranda Christ, Sam Gunn, and Or Zamir. 2024. Undetectable watermarks for language models. In *Proceedings of Thirty Seventh Conference on Learning Theory*, volume 247 of *Proceedings of Machine Learning Research*, pages 1125–1139. PMLR.

Falcon Dai and Zheng Cai. 2019. Towards near-imperceptible steganographic text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4303–4308, Florence, Italy. Association for Computational Linguistics.

Jinyang Ding, Kejiang Chen, Yaofei Wang, Na Zhao, Weiming Zhang, and Nenghai Yu. 2023. Discop: Provably secure steganography in practice based on "distribution copies". In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 2238–2255.

A.A. Fedotov, P. Harremoes, and F. Topsoe. 2003. Refinements of pinsker's inequality. *IEEE Transactions on Information Theory*, 49(6):1491–1498.

Pierre Fernandez, Antoine Chaffin, Karim Tit, Vivien Chappelier, and Teddy Furon. 2023. Three bricks to consolidate watermarks for large language models. In *2023 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6.

Jiayi Fu, Xuandong Zhao, Ruihan Yang, Yuansen Zhang, Jiangjie Chen, and Yanghua Xiao. 2024. Gumbel-Soft: Diversified language model watermarking via the GumbelMax-trick. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5791–5808, Bangkok, Thailand. Association for Computational Linguistics.

Kazuki Fujii, Taishi Nakamura, Mengsay Loem, Hiroki Iida, Masanari Ohi, Kakeru Hattori, Hirai Shota,

[18] The incorrectness of steganographic extraction should refer to text-level inconsistency rates shown in Table 1, which could be higher than 20%.

[19] The token-level inconsistency rates are often below 0.5% shown in Table 2.

Sakae Mizuki, Rio Yokota, and Naoaki Okazaki. 2024. Continual pre-training for cross-lingual llm adaptation: Enhancing japanese language capabilities. In *Proceedings of the First Conference on Language Modeling*, COLM, page (to appear), University of Pennsylvania, USA.

Jonas Geiping, Alex Stein, Manli Shu, Khalid Saifullah, Yuxin Wen, and Tom Goldstein. 2024. Coercing LLMs to do and reveal (almost) anything. In *ICLR 2024 Workshop on Secure and Trustworthy Large Language Models*.

Te Sun Han. 2005. Folklore in source coding: information-spectrum approach. *IEEE Trans. Inf. Theor.*, 51(2):747–753.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization. *Preprint*, arXiv:1412.6980.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 17061–17084. PMLR.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Manli Shu, Khalid Saifullah, Kezhi Kong, Kasun Fernando, Aniruddha Saha, Micah Goldblum, and Tom Goldstein. 2024. On the reliability of watermarks for large language models. In *The Twelfth International Conference on Learning Representations*.

Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. 2024. Robust distortion-free watermarks for language models. *Transactions on Machine Learning Research*.

Sander Land and Max Bartolo. 2024. Fishing for magikarp: Automatically detecting under-trained tokens in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11631–11646, Miami, Florida, USA. Association for Computational Linguistics.

Yuxi Li, Yi Liu, Gelei Deng, Ying Zhang, Wenjia Song, Ling Shi, Kailong Wang, Yuekang Li, Yang Liu, and Haoyu Wang. 2024. Glitch tokens in large language models: Categorization taxonomy and effective detection. *Proc. ACM Softw. Eng.*, 1(FSE).

Aiwei Liu, Leyi Pan, Yijian Lu, Jingjing Li, Xuming Hu, Xi Zhang, Lijie Wen, Irwin King, Hui Xiong, and Philip Yu. 2024. A survey of text watermarking in the era of large language models. *ACM Comput. Surv.*, 57(2).

Yijian Lu, Aiwei Liu, Dianzhi Yu, Jingjing Li, and Irwin King. 2024. An entropy-based text watermarking detection method. *arXiv preprint arXiv:2403.13485*.

Yisroel Mirsky, Ambra Demontis, Jaidip Kotak, Ram Shankar, Deng Gelei, Liu Yang, Xiangyu Zhang, Maura Pintor, Wenke Lee, Yuval Elovici, and Battista Biggio. 2023. The threat of offensive ai to organizations. *Computers & Security*, 124:103006.

Jumon Nozaki and Yugo Murawaki. 2022. Addressing segmentation ambiguity in neural linguistic steganography. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 109–116, Online only. Association for Computational Linguistics.

Naoaki Okazaki, Kakeru Hattori, Hirai Shota, Hiroki Iida, Masanari Ohi, Kazuki Fujii, Taishi Nakamura, Mengsay Loem, Rio Yokota, and Sakae Mizuki. 2024. Building a large japanese web corpus for large language models. In *Proceedings of the First Conference on Language Modeling*, COLM, page (to appear), University of Pennsylvania, USA.

Yuang Qi, Kejiang Chen, Kai Zeng, Weiming Zhang, and Nenghai Yu. 2025. Provably secure disambiguating neural linguistic steganography. *IEEE Transactions on Dependable and Secure Computing*, 22(3):2430–2442.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Jiaming Shen, Heng Ji, and Jiawei Han. 2020. Near-imperceptible neural linguistic steganography via self-adjusting arithmetic coding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 303–313, Online. Association for Computational Linguistics.

Kaiser Sun, Peng Qi, Yuhao Zhang, Lan Liu, William Wang, and Zhiheng Huang. 2023. Tokenization consistency matters for generative models on extractive NLP tasks. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 13300–13310, Singapore. Association for Computational Linguistics.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Dixuan Wang, Yanda Li, Junyuan Jiang, Zepeng Ding, Guochao Jiang, Jiaqing Liang, and Deqing Yang. 2024. Tokenization matters! degrading large language models through challenging their tokenization. *Preprint*, arXiv:2405.17067.

Zihui Wu, Haichang Gao, Ping Wang, Shudong Zhang, Zhaoxiang Liu, and Shiguo Lian. 2024. GlitchMiner: Mining glitch tokens in large language models via gradient-based discrete optimization. *Preprint*, arXiv:2410.15052.

Lingyun Xiang, Shuanghui Yang, Yuhang Liu, Qian Li, and Chengzhang Zhu. 2020. Novel linguistic steganography based on character-level text generation. *Mathematics*, 8(9).

Ruiyi Yan, Tian Song, and Yating Yang. 2024a. A near-imperceptible disambiguating approach via verification for generative linguistic steganography. In *2024 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1638–1643.

Ruiyi Yan, Tian Song, and Yating Yang. 2024b. TokenFree: A tokenization-free generative linguistic steganographic approach with enhanced imperceptibility. In *2024 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 449–455.

Ruiyi Yan, Yating Yang, and Tian Song. 2023. A secure and disambiguating approach for generative linguistic steganography. *IEEE Signal Processing Letters*, 30:1047–1051.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Zhong-Liang Yang, Xiao-Qing Guo, Zi-Ming Chen, Yong-Feng Huang, and Yu-Jin Zhang. 2019. RNN-Stega: Linguistic steganography based on recurrent neural networks. *IEEE Transactions on Information Forensics and Security*, 14(5):1280–1295.

KiYoon Yoo, Wonhyuk Ahn, and Nojun Kwak. 2024. Advancing beyond identification: Multi-bit watermark for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4031–4055, Mexico City, Mexico. Association for Computational Linguistics.

Zhibo Zhang, Wuxia Bai, Yuxi Li, Mark Huasong Meng, Kailong Wang, Ling Shi, Li Li, Jun Wang, and Haoyu Wang. 2024. GlitchProber: Advancing effective detection and mitigation of glitch tokens in large language models. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, ASE '24, page 643–655, New York, NY, USA. Association for Computing Machinery.

Xuandong Zhao, Prabhanjan Vijendra Ananth, Lei Li, and Yu-Xiang Wang. 2024. Provable robust watermarking for AI-generated text. In *The Twelfth International Conference on Learning Representations*.

Zachary Ziegler, Yuntian Deng, and Alexander Rush. 2019. Neural linguistic steganography. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1210–1215, Hong Kong, China. Association for Computational Linguistics.

A Preliminaries & Related Work

A.1 Language Model Basics

A language model (LM) has a vocabulary $\mathcal{V}$ containing words or word fragments known as "tokens." Consider a sequence of LM-generated T tokens $\{s^{(t)}\} \in \mathcal{V}^T$. Entries with negative indices, $[s^{(-N_p)}, \ldots, s^{(-1)}]$, represent a "prompt" of length N_p and $[s^{(0)}, \ldots, s^{(T)}]$ are tokens generated by an LM in response to the prompt.

An LM for the next token prediction at position t, is a function $f_{\mathrm{LM}}(\cdot)$ whose input is a sequence of known tokens $[s^{(-N_p)}, \ldots, s^{(t-1)}]$ which consists of a prompt and the first $t-1$ LM-generated tokens. Then it outputs a logit vector, corresponding to each token in $\mathcal{V}$. These logits are then converted into a discrete probability distribution $\boldsymbol{p}^{(t)} = (p_1^{(t)}, \ldots, p_{|\mathcal{V}|}^{(t)})$ over the vocabulary, by a $\mathrm{softmax}$ operator (for example). The next token is then sampled from $\boldsymbol{p}^{(t)}$ using either standard multinomial sampling, beam search, or greedy sampling and so on.

A.2 Steganography based on Language Models

Alice (the sender) wants to communicate a secret message $m_s \sim U(\{0,1\}^L)$ with Bob (the receiver) by embedding it in a natural language cover text t_s (a stegotext). The uniform distribution is chosen for m_s without loss of generality: if m_s has additional structure it can be further compressed to a uniformly distributed random variable (Han, 2005). Alice and Bob have agreed on an embedding function $\mathcal{S}_{emb}$ and an extracting function $\mathcal{S}_{ext}$ that perform steganography. Alice and Bob also have access to the exact same language model, $\mathcal{M}^o$, which can be used during embedding and extraction. These two functions are supposed to be invertible. In other words, $\mathcal{S}_{emb}(\mathcal{M}^o, m_s) = t_s$, $\mathcal{S}_{ext}(\mathcal{M}^o, t_s) = m_s'$, and m_s' should be equal to m_s.

Generative linguistic steganography utilizes redundancy of candidate pools to achieve steganography. Through further sampling (e.g. top-k) and encoding $\boldsymbol{p}^{(t)}$ with Huffman coding (Yang et al., 2019) or arithmetic coding (Ziegler et al., 2019) and so on, a steganographic candidate pool $\hat{\boldsymbol{c}}^{(t)}$ is obtained, with its probability distribution $\hat{\boldsymbol{p}}^{(t)}$. During embedding process, the language model in turn chooses a token in $\hat{\boldsymbol{c}}^{(t)}$ $(t = 0, 1, \ldots)$ until it encodes the whole secret message m_s; during extraction process, the language model in turn chooses and extracts a token in $\hat{\boldsymbol{c}}^{(t)}$ $(t = 0, 1, \ldots)$ till the end of the stegotext.

A.2.1 Segmentation Ambiguity

The stegotext generated by $\mathcal{S}_{emb}$ is essentially a sequence composed of tokens. The sender must detokenize it using a tokenizer into a stegotext before transmission. As shown in Figure 1a (which is also an illustration of segmentation ambiguity), if the sender generates a token mapping to "_no" and "body", the sender needs to detokenize them into the text "nobody" before sending it to Bob. However, the issue is that common words like "_nobody" often exist as independent tokens "_no" in the model's vocabulary as well. As a result, a single piece of text can correspond to two or even more different token representations. Therefore, during extraction $\mathcal{S}_{ext}(\mathcal{M}^o, t_s)$, since both "_nobody" and "_no" exist in the candidate pool, Bob cannot determine which token the sender embedded the message into. This phenomenon is referred to as *segmentation ambiguity*. This issue can be exempted in only a few tokenizer-free linguistic steganographic approaches (Xiang et al., 2020; Yan et al., 2024b).

A.2.2 Disambiguation Algorithms

Recently, several solutions have emerged to address segmentation ambiguity which achieves 100% disambiguation in extraction.

1) Basic Solution: Nozaki and Murawaki (2022) proposed a simple disambiguation approach, which removes tokens whose mapping subwords are prefixes of others during every generation and extraction step. This process ensures that any token sent by the sender is uniquely extractable for the receiver.

2) MWIS-based Solution: Yan et al. (2023) considered the influence of removing candidate words on the probability distributions and decided to process only if candidate-level ambiguity occurred. Their solution identifies the maximum weight independent set (MWIS) in the candidate pool to reduce probability distortion.

3) SyncPool Solution: Qi et al. (2025) designed provably secure disambiguation linguistic steganography based on ambiguity pool grouping and synchronous sampling to address information loss and token synchronization issues during steganography, eliminating segmentation ambiguity without altering the distribution.

All of these previous disambiguation approaches make the steganographic extraction merely based on prefixes of stegotexts instead of the tokenizer, bypassing the TI between the sender-receiver pair.[20] As a result, these traditional disambiguation methods overly process candidate pools, compromising imperceptibility or embedding capacity.

A.2.3 Imperceptibility of LM-based Steganography

Following the previous formulation (Dai and Cai, 2019; Shen et al., 2020), statistical imperceptibility refers to the similarity between the true language model $\mathcal{M}^t$ in the monitored channel and $\mathcal{M}^s$ which is the language model $\mathcal{M}^o$ integrated with steganographic algorithms. Specifically, the total variation distance (TVD) is used to measure statistical imperceptibility. Consider the TVD between $\mathcal{M}^t$ and $\mathcal{M}^s$, i.e. $d(\mathcal{M}^t, \mathcal{M}^s)$, by triangle inequality:

$$d(\mathcal{M}^t, \mathcal{M}^s) \leq d(\mathcal{M}^t, \mathcal{M}^o), d(\mathcal{M}^o, \mathcal{M}^s) \quad (1)$$

As $d(\mathcal{M}^t, \mathcal{M}^o)$ is a criterion to measure the original language model, which is limited by the research on language models. Thus, $d(\mathcal{M}^o, \mathcal{M}^s)$ is the main focus of linguistic steganography.

According to Pinsker's inequality (Fedotov et al., 2003) and additivity of KL divergence, $d(\mathcal{M}^o, \mathcal{M}^s)$ can be further decomposed in each step, that is:[21]

$$d(\mathcal{M}^o, \mathcal{M}^s) \leq \sqrt{\frac{\ln 2}{2} \sum_{t=1}^{\infty} D_{KL}(\boldsymbol{p}^{(t)} || \hat{\boldsymbol{p}}^{(t)})} \quad (2)$$

where $\boldsymbol{p}^{(t)}$ is the original probability distribution at t^{th} step, and $\hat{\boldsymbol{p}}^{(t)}$ is transformed from $\boldsymbol{p}^{(t)}$ via sampling and encoding. Hence, linguistic steganography could aim to minimize $D_{KL}(\boldsymbol{p}^{(t)} || \hat{\boldsymbol{p}}^{(t)})$, in order to obtain relative near-imperceptibility.

A.3 Watermarks for Language Models

A watermarking algorithm for language models typically comprises two components: a watermark embedding function $\mathcal{W}_{emb}$ and a watermark detecting function $\mathcal{W}_{det}$ (Liu et al., 2024). $\mathcal{W}_{emb}$ takes a language model $\mathcal{M}^o$ and a watermark message m_w as input and outputs a watermarked text t_w, expressed as $\mathcal{W}_{emb}(\mathcal{M}^o, m_w) = t_w$. For the detecting function $\mathcal{W}_{det}$, its input is any text t, and its output is its predicted watermark message for the text, denoted $\mathcal{W}_{det} = m'_w$. The watermark message m_w or m'_w can be a Boolean value (True or False) for zero-bit watermarks to indicate whether the text is generated by AI (Kirchenbauer et al., 2023, 2024), and can also be a bit stream for multi-bit watermark usage (Yoo et al., 2024). So far, there are two main types of inference-time watermarking algorithms: (1) logit-based watermarking and (2) sampling-based watermarking.

For the former, those methods refer to inserting m_w into the logit of each generative step by language models (Kirchenbauer et al., 2023; Fernandez et al., 2023; Kirchenbauer et al., 2024; Lu et al., 2024; Zhao et al., 2024). The trade-off between text quality and detectability should be considered in these watermarks.

For the latter, they do not alter the logits, but utilize the watermark message to guide the sampling process (Aaronson and Kirchner, 2023; Christ et al., 2024; Kuditipudi et al., 2024). For token-by-token sampling watermarking, the principle of incorporating watermarks during the token-sampling phase is derived from the randomness inherent in token sampling. In this scenario, watermarks can be introduced using a fixed seed, where a pseudo-random number generator produces a sequence of pseudo-random numbers to guide the sampling of each token. For watermark detection, it is only necessary to assess the alignment between the text tokens and the pseudo-random numbers, specifically evaluating whether the choice of each token in the text matches the corresponding value in the random number sequence.

A.4 Related Work on Abnormal Tokens

Tokenization stands as a cornerstone in natural language processing, which transforms a continuous text sequence into a list of discrete values called tokens (Wang et al., 2024).

A.4.1 Glitch Tokens

Glitch tokens refer to a class of anomalous tokens in LLMs that can trigger unexpected and often erroneous behaviors when processed by LLMs. This issue arises from improper tokenization of raw texts, which can stem from irregularities in the training process, such as underrepresentation in training data or inconsistencies in tokenization (Geiping et al., 2024).

Glitch token and according glitchy phenomena in LLMs are first investigated comprehensively and

[20] Another disambiguation method (Yan et al., 2024a) is not introduced in this section or adopted as the baseline in experiments, as its disambiguation is reported to be not 100%.

[21] Some derivation is omitted here, as details are verified in (Dai and Cai, 2019; Shen et al., 2020; Fedotov et al., 2003).

systematically by Li et al. (2024), where glitch-token symptoms and glitch-token taxonomy are explored, and an efficient glitch-token detection method named GlitchHunter is proposed. GlitchHunter iteratively constructs a token embedding graph and generates candidate glitch token clusters for subsequent detection.

A more advanced and effective detection and mitigation of glitch tokens is proposed and named GlitchProber (Zhang et al., 2024). This work first reveals the characteristic features induced by glitch tokens on LLMs, which are evidenced by significant deviations in the distributions of attention patterns and dynamic information from intermediate model layers. GlitchProber utilizes small-scale sampling, principal component analysis for accelerated feature extraction, and a simple classifier for efficient vocabulary screening.

Another advancing glitch-token detection method is named GlitchMiner (Wu et al., 2024), which is a gradient-based discrete optimization framework that efficiently identifies glitch tokens by introducing entropy as a measure of prediction uncertainty and employing a local search strategy to explore the token space.

A.4.2 Unreachable Tokens

Besides, 'unreachable tokens' are termed by Land and Bartolo (2024), referring to those tokens that are never produced as a result of tokenizing text. In that work, they test this by checking if decoding a token to a string, and re-tokenizing this string, results in the original token ID. Although they also apply the detokenization-retokenization pipeline, they merely consider that one tested token without contexts.

A.4.3 Tokenization Inconsistency (TI)

The importance of tokenization consistency is reported in extractive NLP tasks (Sun et al., 2023). They study the issue of tokenization inconsistency that is commonly neglected in training these models, and reveal that this issue damages the extractive nature of these tasks after the input and output are tokenized inconsistently by the tokenizer, thus leading to performance drop as well as hallucination.

Besides, a recent work (Wang et al., 2024) constructs an adversarial dataset, named as **ADT** (Adversarial Dataset for Tokenizer), which draws upon the vocabularies of various open-source LLMs to challenge LLMs' tokenization. That study is the first to investigating LLMs' vulnerability in terms of challenging their token segmentation, which will shed light on the subsequent research of improving LLMs' capabilities through optimizing their tokenization process and algorithms.

Absence of SITs					
Generated token ids	...	18	76	...	...
Retokenized token ids	...	18	325	76	...
Absence of CITs					
Generated token ids	...	1092	8	92	...
Retokenized token ids	...	1092	92	...	...

Table 9: Examples of the absence of SITs or CITs when TI occurs.

Correct or consistent tokenization is often overlooked in most tasks. However, in text-based transmission systems (including steganography and watermarking) where texts are transmitted from Alice to Bob, tokenization consistency becomes crucial, as precise transmission is essential for maintaining the integrity of the information.

B Analysis of Inconsistent Tokens and Tokenization Inconsistency (TI)

In this section, we provide detailed analysis and explanations of the relationships between source inconsistent tokens (SITs), consequential inconsistent tokens (CITs), candidate-level inconsistent tokens (candidate-level ITs), and tokenization inconsistency (TI).

Proposition 1. *The sufficient condition for the existence of TI is the existence of SIT(s) or CIT(s).*

According to Proposition 1, when TI occurs, there could be only SITs (in the generated token list) or only CITs (in the retokenized token list). Table 9 provides examples of TI where SITs or CITs are absent. Inconsistent tokens are marked in red background, and other tokens including '...' are all consistent tokens. The essential reason for it is related to tokenizer preferences. For example, the sole SIT or sole CIT could be detokenized to a 0-length character.

Proposition 2. *A necessary condition for the existence of TI is outputting candidate-level IT(s).*

According to Algorithm 1, if outputting a candidate token changes the tokenization state from consistency to inconsistency or persists TI, that candidate token is a candidate-level IT. Therefore, an easy proof of Proposition 2 by contradiction is as follows: *If a candidate-level IT is never generated, TI never occurs.*

Algorithm 2 Stepwise verification (embedding)

Input:
Prompt (initial historical tokens), $[s^{(-N_p)}, \ldots, s^{(-1)}]$
Secret message, m_s
Output:
Steganographic text, t_s

```
1: for $t = 0, 1, \ldots$ do
2:     Apply the language model to historical tokens to obtain the probability distribution $\boldsymbol{p}^{(t)}$ over the vocabulary $\mathcal{V}$.
3:     Sample $\mathcal{V}$ according to $\boldsymbol{p}^{(t)}$ to get the candidate pool $\hat{\boldsymbol{c}}^{(t)}$.
4:     Filter out candidate-level ITs in $\hat{\boldsymbol{c}}^{(t)}$ to get a inconsistency-free candidate pool $\hat{\boldsymbol{c}}'^{(t)}$.
5:     if $\hat{\boldsymbol{c}}'^{(t)} == \emptyset$ then
6:         Add the highest probability token (which is not an SIT) from $\mathcal{V}_{\backslash \hat{\boldsymbol{c}}^{(t)}}$ to $\hat{\boldsymbol{c}}'^{(t)}$.
7:     Get the normalized probability distribution $\hat{\boldsymbol{p}}'^{(t)}$ over $\hat{\boldsymbol{c}}'^{(t)}$.
8:     Use the steganographic embedding algorithm and $m_s$ to generate the next token $s^{(t)}$.
9: Detokenize historical tokens to $t_s$;
10: return $t_s$
```

Proposition 3. *If all the inconsistent tokens are not temporary, there is still possibility that a candidate-level IT does not become an SIT.*

According to Proposition 2, outputting candidate-level ITs are necessary for TI, and according to Proposition 1, SITs are not necessary for TI. Therefore, there are some TI cases where candidate-level ITs are output, but SITs are absent. TI with the absence of SITs (Table 9) provides an example of Proposition 3, where the ‘id: 18’ token or the ‘id: 76’ token should an output candidate-level IT, but neither of them is an SIT.

C Algorithms of Methods

C.1 Stepwise Verification

Algorithm 2 provides details of the steganographic embedding process equipped with our proposed stepwise verification method. This algorithm considers an error scenario with a very small probability of occurrence, that is, the inconsistency-free candidate pool $\hat{\boldsymbol{c}}'^{(t)}$ is $\emptyset$ (Line 4-5). Once it occurs, a non-SIT token outside the steganographic candidate pool $\boldsymbol{c}'^{(t)}$ should be added to $\hat{\boldsymbol{c}}'^{(t)}$, to make sure the generation is always able to continue (Line 6). Algorithm 3 provides the details of the extraction version, and also includes the error prevention mechanism (Line 6-7). At each step of generation, both in embedding and extraction, they share the same view of how the candidate pool is processed.

Algorithm 3 Stepwise verification (extraction)

Input:
Prompt (initial historical tokens), $[s^{(-N_p)}, \ldots, s^{(-1)}]$
Steganographic text, t_s
Output:
Secret message, m_s

```
1: Tokenize $t_s$ to token list $[s^{(-N_p)}, \ldots, s^{(0)}, s^{(1)}, \ldots]$.
2: for $t = 0, 1, \ldots$ do
3:     Apply the language model to historical tokens to obtain the probability distribution $\boldsymbol{p}^{(t)}$ over the vocabulary $\mathcal{V}$.
4:     Sample $\mathcal{V}$ according to $\boldsymbol{p}^{(t)}$ to get the candidate pool $\hat{\boldsymbol{c}}^{(t)}$.
5:     Filter out candidate-level ITs in $\hat{\boldsymbol{c}}^{(t)}$ to get a inconsistency-free candidate pool $\hat{\boldsymbol{c}}'^{(t)}$.
6:     if $\hat{\boldsymbol{c}}'^{(t)} == \emptyset$ then
7:         Add the highest probability token (which is not an SIT) from $\mathcal{V}_{\backslash \hat{\boldsymbol{c}}^{(t)}}$ to $\hat{\boldsymbol{c}}'^{(t)}$.
8:     Get the normalized probability distribution $\hat{\boldsymbol{p}}'^{(t)}$ over $\hat{\boldsymbol{c}}'^{(t)}$.
9:     Use the steganographic extraction algorithm and $s^{(t)}$ to update $m_s$.
10:    Append $s^{(t)}$ to the historical tokens;
11: return $m_s$
```

Algorithm 4 Post-hoc rollback

Input:
Prompt (initial historical tokens), $[s^{(-N_p)}, \ldots, s^{(-1)}]$
Watermark message, m_w
Observation period parameter, q
Output:
Watermarked text, t_w

```
1: $q_c \leftarrow$ NULL;
   /* Initialize the state of observation period*/
2: for $T = 0, 1, \ldots$ do
3:     Apply the language model to historical tokens and watermark embedding algorithm to generate the next token $s^{(t)}$;
4:     Append $s^{(t)}$ to historical tokens;
5:     if Tokenization consistency then
6:         $q_c \leftarrow$ NULL;
7:     else
8:         if $q_c ==$ NULL then
9:             $q_c \leftarrow 0$;
10:        if $q_c \neq$ NULL then
11:            if $q_c < q$ then
12:                $q_c \leftarrow q_c + 1$;
13:            else
14:                Delete the latest $(q + 1)$ historical tokens;
15: Detokenize historical tokens to $t_w$;
16: return $t_w$
```

C.2 Post-Hoc Rollback

Algorithm 4 provides details on how to implement our proposed post-hoc rollback method in the generation process, meanwhile embedding watermarking. q_c is a signal of whether the generation is currently in the state of TI ($q_c ==$ NULL indicates tokenization consistency). Whenever tokenization consistency is recovered, q_c is reset as NULL (Line

5-6). Once the tokenization state changes from consistency to inconsistency, q_c is set as 0 (Line 9), and q_c increases when the inconsistency lasts after generating the next token (Line 12). Once q_c is not less than the designated observation period parameter q, the rollback mechanism is triggered: The token-by-token generation rollbacks back to the state where the stable inconsistent token q tokens ago is not generated (Line 14).

D Experimental Details

D.1 Overall Setups

The initial contexts are randomly selected from the multilingual C4 dataset.[22] The temperature parameter is set to 1.0 constantly. According to the features of different languages, for Llama-2-7b, the initial 10 words of an item in the C4 dataset are the initial context for each generation; while for Swallow-7b and Qwen2.5-7b, the initial 10 characters of an item in the C4 dataset are the initial context for each generation. The perplexity of a text is calculated by the language model that generates the text. All the parameters of the tokenizer functions are default, except for setting `skip_special_tokens = True` in detokenization.

All experiments are implemented in Python 3.12.7 with Torch 2.5.0, running on a 2.0 GHz CPU and accelerated by using 8 × NVIDIA RTX A6000 GPUs.

D.2 Steganalysis

As a discriminator for each language, we used a base-sized BERT model taken from Hugging Face's transformers package (English: bert-base-uncased,[23] Japanese: cl-tohoku/bert-japanese,[24] Chinese: bert-base-chinese).[25] Positive samples are collected from stegotexts generated using various top-k samplings, while negative samples are sourced from non-steganographic texts (generated by the same models without any steganographic algorithm).

As for each top-k sampling value ($k \in \{4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096\}$ - 11 different k), for each disambiguation method and ours (4 methods), and for each language model, 500 samples are generated, for each language model the size of collected stegotexts is $11 \times 4 \times 500 = 22000$. Hence, for each language model, during the training phase, both positive and negative samples consist of 17,600 instances each (80% of all collected stegotexts). For testing, 4,400 untrained positive samples are used (20% of all collected stegotexts), categorized into different embedding-capacity intervals as shown in Tables 5, 6, and 7. In each embedding-capacity interval and for each disambiguation approach, only stegotexts with a sample size greater than 20 are included in the tests; otherwise, "–" is marked to indicate insufficient data.

Given the significant variation in the lengths of positive samples, we adjust the negative samples to uniformly vary between 10 and 128 tokens (the prompt is excluded) to ensure that the trained discriminator is not sensitive to text length. Additionally, all texts are padded or truncated to 128 tokens, so that positive samples cannot be distinguished as steganographic based solely on their length. For fine-tuning the BERT model, we use Adam (Kingma and Ba, 2017) as the optimizer with a learning rate of 5×10^{-5}. The batch size is set to 2048, and the discriminator is trained for 20 epochs, running time of the whole training process is approximately 10 minutes.

	Llama-2-7b		**Swallow-7b**		**Qwen2.5-7b**	
k	**BPT**	**ER**	**BPT**	**ER**	**BPT**	**ER**
4	1.11	20.0%	0.95	6.0%	1.01	14.0%
8	1.62	14.6%	1.38	5.0%	1.45	6.8%
16	2.03	11.4%	1.77	4.0%	1.98	7.4%
32	2.34	8.0%	2.12	4.4%	2.37	6.2%
64	2.64	9.4%	2.45	3.2%	2.77	8.8%
128	2.87	8.0%	2.68	5.2%	3.08	7.2%
256	3.08	9.2%	2.94	3.6%	3.43	7.4%
512	3.21	11.4%	3.23	4.6%	3.71	6.8%
1024	3.27	7.2%	3.38	4.6%	4.05	8.0%
2048	3.41	6.2%	3.71	5.4%	4.25	7.6%
4096	3.41	11.4%	3.69	3.4%	4.36	7.6%

Table 10: Embedding capacities and error rates of steganography (without disambiguation or stepwise verification) implemented under various language models and top-k sampling values.

D.3 Original Error Rates

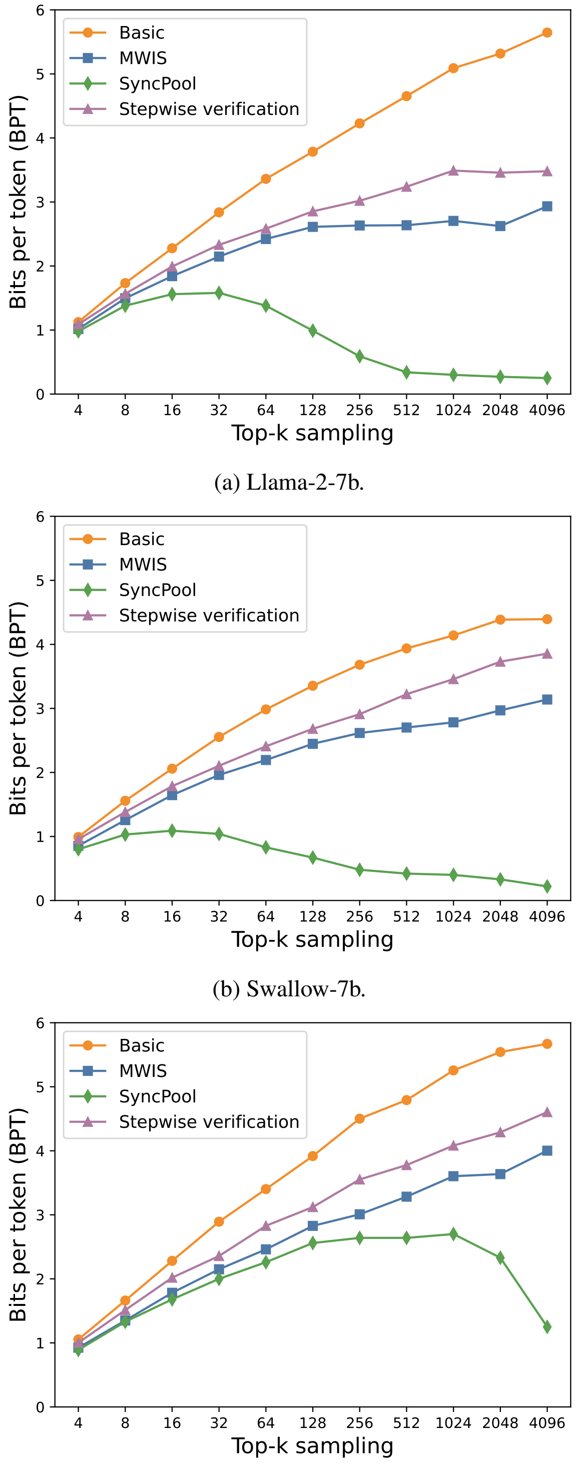
In this section, we use empirical statistics to show the extent to which steganography suffers from extraction errors, if extraction errors are neglected in steganographic approaches. We use the steganographic extraction error rate (**ER**) to indicate the rate that incorrect extraction occurs, and for approaches equipped with neither disambiguation nor our proposed stepwise verification, the error rates

[22] https://huggingface.co/datasets/allenai/c4

[23] https://huggingface.co/google-bert/bert-base-uncased

[24] https://github.com/cl-tohoku/bert-japanese

[25] https://huggingface.co/google-bert/bert-base-chinese



(a) Llama-2-7b.

(b) Swallow-7b.

(c) Qwen2.5-7b.

Figure 5: Average embedding capacities (bits per token, BPT) when using various disambiguation methods and our stepwise verification method, under top-k values.

are referred to as original error rates. Table 10 lists **BPT** and **ER** for steganography with extraction errors. For each language model and for each top-k value, the data size is 500. For $k = 4$, **BPT** is the lowest and **ER** is the highest for each language model. The main reason is that when embedding a 128-bit secret message, lower **ER** means longer generated stegotext. And according to the relationship between the token number and the text-level inconsistency rate shown in Table 1, it is reasonable for longer stegotexts to suffer from higher **ER**. In addition, when the length of the secret message increases, it is reasonable to anticipate that the original error rates for various language models increase further.

D.4 The Limitation of SyncPool in Embedding Capacity

Figure 5 illustrates how embedding capacity varies according to various top-k values when different methods are adopted. For all three language models with respectively English, Japanese and Chinese contexts, these data points in SyncPool share a similar trajectory, that is, when top-k value increases, **BPT** increases when k is small and decreases when k becomes much larger. For comparison, in other methods, when top-k value increases, **BPT** increases steadily.

The reason for the phenomenon is that SyncPool merges the original candidate pools into ambiguous pools for subsequent steganographic processing. However, as the ambiguous pools are formed according to prefix relationships in candidate tokens, when the size of original candidate pools increases, the average tokens in each ambiguous pool also increase. As a result, the size of ambiguous pools could rise more rapidly than the size of original candidate pools, so the average number of ambiguous pools could decrease, thus leading to lower embedding capacity.

Furthermore, the work of SyncPool (Qi et al., 2025) reports a KL divergence of 0 in their experiments, as their reference candidate pools are based on top-k sampled candidates. However, as outlined in Eq. 2, to accurately compare the divergence between the original language model and the model used for steganography, we compute **KLD** using the original candidate pools as references in our experiments (Tables 5, 6, and 7).

D.5 Watermarking Schemes

Consider a text $[s^{(1)}, ..., s^{(T)}]$, its watermark strength (an indicator of detectability) is denoted as $\Phi(s^{(1)}, ..., s^{(T)})$.

For logit-based watermarking, including LeftHash (Kirchenbauer et al., 2023), SelfHash (Kirchenbauer et al., 2024) and Unigram (Zhao et al., 2024) adopted by our experiments, their mechanisms are as follows:

- **Context:** For LeftHash and SelfHash, the context is previous h tokens; there is no context for Unigram.
- **Pseudo-random function:** For LeftHash and SelfHash, $F_{sk}(\text{context})$ hashes the context to a seed at each generative step; Unigram adopts a global seed. Then the seed is used to generate a random vector $\text{Vec}^{(t)}$ in $\{0,1\}^{|\mathcal{V}|}$, the vector has $\gamma|\mathcal{V}|$ 1's (representing green tokens) and $(1-\gamma)|\mathcal{V}|$ 0's (representing red tokens).
- **Decoder:** Sample a token $s^{(t)}$ from $\text{softmax}(\text{logit}^{(t)} + \delta * \text{Vec}^{(t)})$.
- **One-token score:** $\phi^{(t)} = \text{Vec}^{(t)}[s^{(t)}]$.
- **Watermark strength:** $\Phi(s^{(1)}, ..., s^{(T)}) = \frac{\sum_{t=1}^{T} \phi^{(t)} - \gamma T}{\sqrt{T\gamma(1-\gamma)}}$.

For sampling-based watermarking, the detailed scheme of Gumbel softmax scheme (Aaronson and Kirchner, 2023) is:

- **Context:** The previous h tokens.
- **Pseudo-random function:** $F_{sk}(\text{context})$ hashes the context to a seed at each generative step, then uses the seed to generate a random vector $\text{Vec}^{(t)}$ in $(0,1)^{|\mathcal{V}|}$ where each element is uniformly sampled from $(0,1)$.
- **Decoder:** Select a token $s^{(t)}$ which is $\text{argmax}_{1\leq i\leq|\mathcal{V}|} \frac{\log(\text{Vec}^{(t)}[i])}{\text{softmax}(\text{logit}^{(t)})[i]}$.
- **One-token score:** $\phi^{(t)} = -\log(1 - \text{Vec}^{(t)}[s^{(t)}])$.
- **Watermark strength:** $\Phi(s^{(1)}, ..., s^{(T)}) = \frac{1}{\sqrt{T}} \sum_{t=1}^{T} \phi^{(t)} - \sqrt{T}$.

D.6 Perplexities of Attacked Watermarked Texts

Tables 11, 12, and 13, respectively, list perplexities of attacked watermarked texts at each attack probability ($\epsilon = 0.2$ or $\epsilon = 0.4$) under three language models. In terms of the perplexity metric, the superiority of addressing TI exists in attacked scenarios as well as unattacked scenarios (shown in Tables 8, 14, and 15).

Watermarking scheme		**Attacked** ($\epsilon = 0.2$)	**Attacked** ($\epsilon = 0.4$)
LeftHash	Original	185.25	477.71
	Post-hoc rollback	**162.56**	**417.41**
SelfHash	Original	188.69	470.96
	Post-hoc rollback	**172.93**	**455.81**
Unigram	Original	167.16	438.53
	Post-hoc rollback	**161.24**	**424.75**
Gumbel	Original	19.38	46.32
	Post-hoc rollback	**18.70**	**40.70**

Table 11: Perplexities in various watermarking schemes under attack scenarios when Llama-2-7b is adopted.

Watermarking scheme		**Attacked** ($\epsilon = 0.2$)	**Attacked** ($\epsilon = 0.4$)
LeftHash	Original	133.44	301.82
	Post-hoc rollback	**130.31**	**290.67**
SelfHash	Original	143.03	**308.02**
	Post-hoc rollback	**142.34**	313.01
Unigram	Original	136.13	301.39
	Post-hoc rollback	**134.44**	**291.96**
Gumbel	Original	5.99	9.44
	Post-hoc rollback	**5.78**	**9.29**

Table 12: Perplexities in various watermarking schemes under attack scenarios when Swallow-7b is adopted.

Watermarking scheme		**Attacked** ($\epsilon = 0.2$)	**Attacked** ($\epsilon = 0.4$)
LeftHash	Original	414.34	962.58
	Post-hoc rollback	**378.68**	**857.41**
SelfHash	Original	398.99	890.30
	Post-hoc rollback	**355.39**	**786.71**
Unigram	Original	**309.83**	686.18
	Post-hoc rollback	322.16	**676.22**
Gumbel	Original	6.96	12.76
	Post-hoc rollback	**6.69**	**11.57**

Table 13: Perplexities in various watermarking schemes under attack scenarios when Qwen2.5-7b is adopted.

E Text Samples

E.1 Samples of Stegotexts

Table 16 presents examples of stegotexts generated by our proposed stepwise verification method. Each generated text embeds a 128-bit random secret message. Following the approach of Ziegler et al. (Ziegler et al., 2019), we terminate the generation process once the proposed method has finished embedding the message.

E.2 Samples of Watermarked Texts

Table 17 lists some examples of watermarked texts generated by our proposed post-hoc rollback method, and as the number of generated tokens is 200 for each text, the complete watermarked texts

Watermarking scheme		**Unattacked**			**Attacked** ($\epsilon = 0.2$)		**Attacked** ($\epsilon = 0.4$)	
		Watermark Strength↑	AUROC↑	PPL↓	Watermark Strength↑	AUROC↑	Watermark Strength↑	AUROC↑
LeftHash	Original	8.05	0.999	20.47	**5.18**	0.987	**3.04**	0.895
	Post-hoc rollback	**8.07**	0.999	**20.23**	5.13	**0.988**	3.02	**0.896**
SelfHash	Original	7.96	0.998	22.21	4.58	0.971	2.54	0.855
	Post-hoc rollback	**7.99**	0.998	**21.72**	**4.67**	**0.975**	**2.58**	**0.859**
Unigram	Original	8.26	0.999	19.80	6.98	0.988	5.54	0.917
	Post-hoc rollback	**8.32**	0.999	**19.74**	**7.02**	0.988	**5.78**	**0.921**
Gumbel	Original	29.97	**0.910**	1.97	22.36	0.885	16.63	0.862
	Post-hoc rollback	**31.24**	0.905	**1.95**	**23.33**	**0.887**	**17.01**	**0.864**

Table 14: Quantitative comparison in various watermarking schemes on Swallow-7b and Japanese context.

Watermarking scheme		**Unattacked**			**Attacked** ($\epsilon = 0.2$)		**Attacked** ($\epsilon = 0.4$)	
		Watermark Strength↑	AUROC↑	PPL↓	Watermark Strength↑	AUROC↑	Watermark Strength↑	AUROC↑
LeftHash	Original	8.47	0.999	62.42	5.36	0.991	2.85	0.901
	Post-hoc rollback	**8.54**	0.999	**58.12**	**5.37**	**0.992**	**3.00**	**0.910**
SelfHash	Original	8.29	1.000	58.00	4.67	0.975	2.13	0.829
	Post-hoc rollback	**8.35**	1.000	**53.16**	**4.68**	**0.978**	**2.22**	**0.835**
Unigram	Original	9.23	1.000	54.10	7.67	0.994	**6.22**	**0.954**
	Post-hoc rollback	**9.28**	1.000	**51.84**	**7.72**	0.994	6.07	0.939
Gumbel	Original	22.09	0.897	2.37	15.96	**0.859**	11.58	0.793
	Post-hoc rollback	**22.46**	**0.900**	2.37	**16.27**	0.858	**12.17**	**0.796**

Table 15: Quantitative comparison in various watermarking schemes on Qwen2.5-7b and Chinese context.

Model	Prompt	Stegotext	BPT	PPL	KLD
Llama-2-7b	Finding a high quality photographer for your family portrait,	Finding a high quality photographer for your family portrait, event or wedding is not as easy as it sounds. A photography business can offer photographers for hire that are affordable, but this does not mean that you have to compromise on quality. Here are more tips to help you find the ultimate business that	2.33	6.23	0.024
Llama-2-7b	I've never understood the whole points thing, scholarships,	I've never understood the whole points thing, scholarships, etc. Now it sounds like a full ride (or nearly so). The deal is still a good one. That probably sounds heartless to the	4.41	26.64	0.009
Swallow-7b	該当する商品が、売り	該当する商品が、売りたい商品の条件に一致している場合に、お勧めできる相手を探して、1社から該当する会社への発送を依頼する方法が出品	3.76	21.99	0.059
Swallow-7b	褐色 仙台八幡、汚れ	褐色 仙台八幡、汚れ再生塗装：仙台 外壁塗装 お客さまの声 外壁塗装終わり、綺麗に汚れが落ちて大満足!仙台中央	4.27	71.62	0.013
Qwen2.5-7b	"全脑开发"真能让孩	"全脑开发"真能让孩成长吗？这个"脑"就是指孩子大脑的开发潜能。全脑开发的主要内容包括情绪、记忆力、思维力、想象力、创造力等智力因素和观察力、注意力、思维力等非智力因素的	2.72	11.39	0.096
Qwen2.5-7b	荷兰国际集团预计美国	荷兰国际集团预计美国五金商品经销商对抵押率为25%的卷心菜/枫叶拱	7.53	233.92	0.027

Table 16: Examples of stegotexts generated with our stepwise verification method.

are not provided due to space limitation.

E.3 Samples of How TI Influences Perplexities

Table 18 provides examples to clarify how consequential inconsistent tokens (CITs) affect the predicted probabilities of each token in the token list

Model	Method	Prompt	Stegotext	Watermark strength	PPL
Llama-2-7b	LeftHash	In fact, there are currently over a million international	In fact, there are currently over a million international students enrolled around the world? These valuable languages, particularly the most important ones, offer infinite possibilities; with them, you'll set out[...continues]	8.19	16.79
Llama-2-7b	SelfHash	So you have your children writing every day. Great!	So you have your children writing every day. Great! But it's important not to think "writing" means that only stories and poems will qualify. In the course of life, children will write in any number of ways:[...continues]	6.82	16.55
Swallow-7b	LeftHash	『秋晴れや 千種若水	『秋晴れや 千種若水 父逝きぬ』 (『敬老の日』) 18年前に亡くした父親のことをのせた俳句です。2003年(平成15年)とイ「ンタネット」がリ1995年(平成7年)より普[...continues]	9.71	32.14
Swallow-7b	Unigram	【札幌（新千歳）発】	【札幌（新千歳）発】 北海道クルーズ旅物語 こんな方にお勧め! (もちろん、リクエスト型をお選びいただいた方に限ります) 札幌観光と北海道を網羅する内容にしたい![...continues]	7.50	14.36
Qwen2.5-7b	LeftHash	书法教育：临摹还是创	书法教育：临摹还是创发我们的书法临写，可以从追溯300年传统书法产生之时就分为帖学和碑学两个主要的传统。行摹也是帖学书法传承的主要途径。但在现实生活[...continues]	8.55	116.27
Qwen2.5-7b	Gumbel	湖南茶博会全省30个	湖南茶博会全省30个市州组团来长沙设馆参展星辰在线5月15日讯5月15日，2023湖南茶叶博览会(简称"茶博会")专业观众招募活动媒体吹风会在长沙举行，[...continues]	4.48	2.76

Table 17: Examples of watermarked generated with our post-hoc rollback method.

Llama-2-7b				
Generated token list	[Previous tokens]...	'eye'	'q'	-
Predicted probability	[Previous probabilities]...	6.59×10^{-5}	2.94×10^{-4}	-
Retokenized token list	[Previous tokens]...	'e'	'y'	'eq'
Predicted probability	[Previous probabilities]...	9.49×10^{-6}	9.79×10^{-7}	9.27×10^{-8}
Swallow-7b				
Generated token list	[Previous tokens]...	'に'	'生'	'え'
Predicted probability	[Previous probabilities]...	1.91×10^{-3}	2.43×10^{-4}	4.53×10^{-4}
Retokenized token list	[Previous tokens]...	'に'	'生え'	-
Predicted probability	[Previous probabilities]...	1.91×10^{-3}	2.17×10^{-6}	-
Qwen2.5-7b				
Generated token list	[Previous tokens]...	'医生'	'态度'	-
Predicted probability	[Previous probabilities]...	4.06×10^{-3}	5.27×10^{-2}	-
Retokenized token list	[Previous tokens]...	'医'	'生态'	'度'
Predicted probability	[Previous probabilities]...	1.94×10^{-2}	1.36×10^{-6}	4.61×10^{-6}

Table 18: Examples of how TI effects predicted probabilities of tokens.

when the TI occurs, where CITs are marked in red background. These examples show the fact that these predicted probabilities of CITs are generally lower than those of SITs, whereas only these CITs can be accessed by perplexity calculation.

The expression of calculating perplexity is:

$$\text{PPL} = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log P(s^{(i)} \mid s^{(1):(i-1)})\right)$$

	Llama-2-7b	Swallow-7b	Qwen2.5-7b
$N = 1$	0.00%	1.36%	60.03%
$N = 2$	3.61%	68.02%	27.69%
$N = 3$	3.61%	2.33%	0.20%
...	...	...	...
$N = \infty$	91.24%	18.02%	12.07%

Table 19: The percentage that temporary inconsistency disappears after generating N tokens afterwards.

where N is the length of the retokenized token list, $s^{(i)}$ denotes the i^{th} token in this list, and $P(s^{(i)} \mid s^{(1):(i-1)})$ represents the predicted probability of $s^{(i)}$ according to historical $i-1$ tokens. Therefore, when the predicted probabilities of CITs are lower than SITs, the resulting perplexity is higher than that if perplexity calculation is based on the original generate token list (which includes SITs).

F Determine Observation Period (q)

Table 20 lists the percentage that temporary inconsistency naturally disappears after generating N tokens afterwards. And $N = \infty$ denotes a permanent inconsistency (in 100-token texts, 1000 samples for each model). According to this table, we can find that, in Swallow-7b and Qwen2.5-7b, most temporary inconsistencies disappear after generating 2 subsequent tokens (because of partial UTF-8 tokens), while in Llama-2-7b, most temporary inconsistencies are much more stable (because of special tokens).

Therefore, back to the principled way to determine, in Swallow-7b and Qwen2.5-7b, is set at least greater than 2 (we set $q = 10$ in experiments) for to avoid most temporary inconsistencies (to avoid the false positive), and in Llama-2-7b, can be set as a much smaller value (we set $q = 2$ in experiments), because most inconsistencies which have happened will not disappear, which means that it is suitable to be fixed immediately.

Table 20 also explains the significant differences in the temporariness of inconsistent tokens between Llama-2-7b and the other two language models, as more than 90% temporary inconsistencies do not disappear in Llama-2-7b (shown in Table 4).

G Efficiency of the Post-Hoc Rollback Method for Watermarking

Table 20 reports the running time (in seconds) for various watermarking schemes across different language models, based on 500 samples with 200 tokens each. The results show that the runtime differences between the original methods and our proposed post-hoc rollback method are minimal — generally under 5% — indicating that our approach introduces minor computational overhead.

H Attacking Watermarks with GPT-4o Paraphrasing

Table 21 lists the average experimental statistics where the watermarked texts are attacked by GPT-4o paraphrasing (`temperature = 1` and `max_completion_tokens = 2048`). Specifically, the prompt template is:

Paraphrasing template of GPT-4o

System message: You are tasked to paraphrase. Please directly paraphrase the text you receive (in the corresponding language).
User prompt: <text>

From Table 21, we can find that our proposed post-hoc rollback method for watermarking overall improves the robustness against paraphrasing attacks based on strong a large language model.

Watermarking scheme		Llama-2-7b	Swallow-7b	Qwen2.5-7b
LeftHash	Original	6.89	7.13	6.03
	Post-hoc rollback	(+1.16%) 6.97	(+3.51%) 7.38	(+3.98%) 6.27
SelfHash	Original	7.04	7.24	5.96
	Post-hoc rollback	(+0.85%) 7.10	(+3.89%) 7.52	(+2.34%) 6.10
Unigram	Original	6.88	7.17	6.11
	Post-hoc rollback	(+2.90%) 7.08	(+5.44%) 7.56	(+5.24%) 6.43
Gumbel	Original	6.85	7.26	6.00
	Post-hoc rollback	(+2.04%) 6.99	(+3.03%) 7.48	(+4.17%) 6.25

Table 20: Running time (seconds) in various watermarking schemes when different language models are adopted.

Watermarking scheme		Llama-2-7b		Swallow-7b		Qwen2.5-7b	
		Watermark Strength ↑	AUROC ↑	Watermark Strength ↑	AUROC ↑	Watermark Strength ↑	AUROC ↑
LeftHash	Original	2.57	0.876	**2.92**	**0.887**	3.47	0.913
	Post-hoc rollback	**2.60**	**0.877**	2.91	0.886	3.47	**0.915**
SelfHash	Original	1.86	0.803	2.35	0.837	2.42	0.861
	Post-hoc rollback	**1.99**	**0.815**	2.35	**0.840**	**2.44**	**0.865**
Unigram	Original	4.56	0.898	6.63	0.946	4.39	0.918
	Post-hoc rollback	**4.59**	**0.905**	**6.69**	**0.950**	**4.51**	**0.923**
Gumbel	Original	10.62	0.887	18.18	0.877	15.00	**0.851**
	Post-hoc rollback	**11.07**	**0.895**	**18.56**	**0.879**	**15.32**	0.850

Table 21: Quantitative comparison in various watermarking schemes under GPT-4o paraphrasing attack when different language models are adopted.